Национальный исследовательский университет информационных технологий,
механики и оптики.
Кафедра Информатики и Прикладной Математики.
Тестирование ПО.

Лабораторная работа №1
11 вариант

Работу выполнил студент группы P3315
Халанский Дмитрий

2016

1. Цели работы

- Изучение принципов разработки unit-тестов;
- Исследование популярных каркасов для unit-тестирования;
- Тестирование реализации алгоритма.

2. Задание

1. Составить unit-тесты к реализации алгоритма параллельной обменной сортировки Бетчера, подтверждающие или опровергающие её корректность;
2. Написать реализацию алгоритма параллельной обменной сортировки Бетчера, проходящую тесты, написанные в ходе выполнения первой части.

3. Ход работы

Для реализации алгоритма был выбран язык Clojure. Он поддерживается платформой CLR и обладает рядом встроенных средств для unit-тестирования.

3.1. Тесты

```
(use 'clojure.test)
(load "bitonic")

(defn lstsorted?
  "Return true if list is sorted"
  [l]
  (if (>= 1 (count l))
    true
    (let [[a b] [(first l) (first (rest l))]]
      (and (<= a b) (lstsorted? (rest l))))))

(deftest sortedtest-emp
  (is (lstsorted? [])))

(deftest sortedtest-true
  (is (lstsorted? [1 2 3 4])))

(deftest sortedtest-false
  (is (not (lstsorted? [1 3 2 4]))))

(deftest emptytest
```

```
  (is (empty? (bitonic-srt true [])))))

(deftest singletest
  (is (= 1 (count (bitonic-srt true [1])))))

(deftest sortedtest
  (is (lstsorted? (bitonic-srt true [1 2 3 4 5 6 7]))))

(deftest reversetest
  (is (lstsorted? (bitonic-srt true [7 6 5 4 3 2 1]))))

(deftest huge
  (is (lstsorted? (bitonic-srt true [45 6 542 5 23 6 2 7 98 23 7 4 3]))))

(deftest really-huge
  (let [lst (take 1000 (repeatedly #(rand-int 50000)))]
    (is (lstsorted? (bitonic-srt true lst)))))

(run-tests)
```

3.2. Реализация

```
(defn greatestP2ltN
  "Finds the greates power of 2 less than n"
  [n] (loop [p2 2] (if (>= p2 n) (/ p2 2) (recur (* 2 p2)))))

(defn bitonic-mrg
  "Merges bitonic sequences"
  [up coll]
  (if (= 1 (count coll))
    coll
    (let [[l1 l2] (split-at (greatestP2ltN (count coll)) coll)]
      (let [[l1' l2']
            (loop [l [] r [] l1 l1 l2 l2]
               (if (empty? l2)
                 [(concat l l1) r]
                 (let [[ln rn]
                       (let [[fl1 fl2] [(first l1) (first l2)]]
                         (if (= (< 0 (compare fl1 fl2)) up)
                           [fl2 fl1]
                           [fl1 fl2]))]
                   (recur (conj l ln) (conj r rn) (rest l1) (rest l2)))))]
        (concat (bitonic-mrg up l1') (bitonic-mrg up l2'))))))
```

```
(defn bitonic-srt
  "Bitonic sort"
  [up coll]
  (if (or (empty? coll) (= 1 (count coll)))
    coll
    (let [[l1 l2] (split-at (/ (count coll) 2) coll)]
      (bitonic-mrg up (concat (bitonic-srt (not up)  l1)
                              (bitonic-srt up        l2))))))
```

4. Выводы

В результате проделанной работы мы ознакомились со средствами unit-тестирования в языке Clojure, написали ряд тестов и проходящую их реализацию алгоритма сортировки.